Fundamentals Of Dependable Computing

ICSE 2014

John C. Knight
Department of Computer Science
University of Virginia
Tutorial Goals

- Answer the following questions:
  - What is dependability and what affects it?
  - How do we achieve dependability?
  - How do we know we have achieved it?
- Learn a few things, have some fun, etc.
- Motivation level, cannot teach several semester’s worth of material in a day
- Assume that fundamental material is unfamiliar
- Emphasis on software

**Dependability — A Rigorous, Principled, Engineering Subject**
Tutorial Roadmap

You are here

Basic Concepts → Terminology Of Dependability → Dependability Requirements

Faults And Fault Treatment → Software Technology Review → Software Fault Avoidance

Software Fault Elimination → Fault Tolerance → Special Topics
Tutorial Schedule

- **Session 1:**
  - Basic concepts
  - Terminology
  - Dependability requirements

- **Session 2:**
  - Faults
  - Fault treatment
  - Software technology review

- **Session 3:**
  - Formal specification
  - Correctness by construction with SPARK Ada

- **Session 4:**
  - Hardware and software fault tolerance
  - Byzantine agreement
  - Fail-stop machines
Tutorial Protocol

- Please call me John
- Please ask questions anytime
- Please relate your experience
- Please ask me to slow down if necessary
- Please indicate material that is:
  - Too high or too low level
  - Needs clarification
  - Wrong
- Make sure that this is fun and useful to you
- Copies omit some slides (copyright)
Tutorial Assumed Background

- High-level language programming
- Basic computer organization
- Elementary probability theory
- Predicate calculus
- Set theory
- Basic principles of operating systems
Session 1
Terminology And Requirements
Starting Point

Entrance to the Meissen Pottery Factory near Dresden in Germany
What Is Our Goal?

A chain is only as strong as its weakest link...

Many things could go wrong:
- Attachment could break
- Box could break
- Chain could break
- Etc.
Dependability Outline

1. Determine what happens if chain/box breaks
   - Determine the consequences of failure
2. Decide how strong the chain/box has to be
   - Define dependability requirements
3. Determine how chain/box might fail
   - Determine all possible faults that might occur
4. Buy high quality chain/box so as to avoid defects
   - Practice fault avoidance during construction
5. Fix any broken links that remain
   - Practice fault elimination during construction
6. Cope with broken links you didn’t find or that break
   - Practice fault tolerance during operation
1. Determine the consequences of failure
2. Define dependability requirements
3. Determine all possible faults that might occur
4. Practice fault avoidance during construction
5. Practice fault elimination during construction
6. Practice fault tolerance during operation
Dependable Computing Systems
What Happened?

- **Titan IV:**
  - Power bus short circuit, guidance reboot
- **Mars Global Explorer:**
  - Unit systems confusion
- **Mars Polar Lander:**
  - Touchdown sensor triggered by landing leg deployment
What Happened?

- Mars Rovers:
  - Pathfinder - Priority inversion
  - Spirit and Opportunity – file system filled up

- Korean 801:
  - Safety case violated

- Ariane V:
  - Unhandled exception
Software And Its Role

Software caused all of these consequences
How To Think About Failures

- Accidents are **very** complex:
  - Do not jump to conclusions
  - There is never a “root” or single cause
  - There are usually **many** causal factors

- Lessons learned must be comprehensive:
  - Prevent future occurrences of all causal factors

- *Always* investigate failures
Role of The Software Engineer

- Why should a software engineer be concerned about dependability?

- Four reasons:
  - Software has to perform the intended function
  - Software has to perform the intended function correctly
  - Software has to operate on target system whose design might have been influenced by system’s dependability goals
  - Software often needs to take action when a hardware component fails
So?

- Acquire sufficient information about systems side of dependability that the software engineering can:
  - Understand why software is being asked to do what it is being asked to do
  - Understand why software is being made to operate on the particular platform specified by the system designers

- Acquire a conceptual framework within which to think about software and its dependability
Current Software Practice

- Unstructured, natural language documents
- Some use of visual specification & formalism from other disciplines (PDE’s)
- UML, C, C++, Java
- Tools for test setup and management
- Combative adoption of standards

GREAT CARE By Many GREAT PEOPLE
Dependable Systems

- Software is the least well understood technology involved
- Software volume is increasing at an exponential rate
- Software is quickly becoming the dominant cause of critical system failures
- Many aspects of hardware dependability are essentially solved problems
Step 1
Determine Consequences of Failure

What Happens If The Chain Breaks?
Consequences of Failure

- Injury or loss of life
- Environmental damage
- Damage to or loss of equipment

Financial loss:
- Theft
- Useless or defective mass-produced equipment
- Loss of production capacity or service
- Loss of business reputation, customer base
Consequences of Failure

- **Direct**—device fails and causes injury
- **Indirect**—defective support tool leads to device that fails

Combinations of losses:
- Immediate damage to equipment
- Subsequent loss of service
- Subsequent loss of business reputation
- Subsequent law suits
Hidden Costs of Failure

- Some costs of failure are obvious, others not
- Ariane V example:
  - **Obvious:**
    - Loss of launch vehicle
    - Loss of payload
  - **Not so obvious:**
    - Environmental cleanup
    - Loss of service from payload
    - Vehicle redesign
    - Increased insurance rates for launches
Consequences of Failure

- Loss of revenue
  - Software defects: $200 billion/year (SCC)
    - Note: recent worms and viruses
  - 1 hour of downtime costs (InternetWeek, 2000):
    Brokerage operations: $6,450,000/hr
    Credit card auth.: $2,600,000/hr
    Ebay (1 outage/22hrs): $225,000/hr
    Home Shopping Channel: $113,000/hr
    Airline reservation ctr: $89,000/hr
    ATM services fee: $14,000/hr

- Note: Ebay (22 hrs, 1999)
  - $4 Billion Market Cap loss
Step 2
Define Dependability Requirements

*How Strong Does The Chain Have To Be?*
Why Dependability Requirements?

- Need an engineering target
  - What technology do we have to employ?
- Informal statements are inadequate:
  - “System should not fail.”
  - “System must be very reliable.”
- Need a target so that, if we meet it, system will perform “acceptably”
- Need terminology to define requirements
Terminology of Dependability
Why Study Terminology?

- We need to be able to communicate in a precise manner:
  - Researchers
  - Developers
  - Customers

- There are everyday notions of these terms

- The public has an interest

- But public terminology is imprecise
Webster

- **Dependable**: capable of being depended on: RELIABLE

- **Reliable**: suitable or fit to be relied on: DEPENDABLE

  - **Rely**:
    1) to be dependent <the system for which we depend on water>
    2) to have confidence based on experience <someone you can rely on>
Historical Evolution of Concerns

- **40’s: ENIAC**
  - 18K vacuum tubes → failed ~ every 7 mins
  - 18K multiplies/minute → 7 mins ~ one program execution
  
  **Need RELIABILITY**

- **60’s: Interactive systems**
  + **AVAILABILITY**

- **70’s: F-8 Crusader, MD-11, Patriot missile defense**
  + **SAFETY**

- **90’s-today: Internet, E-commerce, Grid/Web services**
  + **SECURITY**
We will use this material frequently

For now, we will look at dependability & failure
Old:
Dependability is the ability to deliver service that can justifiably be trusted.

New:
The dependability of a system is the ability to avoid service failures that are more frequent and more severe than is acceptable.

Why the change? To allow for the possibility of failure yet the system remaining acceptable.
Social Aspects

- Wholesale death:
  - Aircraft, railway accidents
  - Improper detonation of weapons

- Retail death:
  - Car accidents, medical devices
  - Operational procedures

- Societal assessment determines acceptable levels of loss
Failure

From the taxonomy:

Correct service is delivered when the service implements the system function. A **service failure**, often abbreviated here to **failure**, is an event that occurs when the delivered service deviates from correct service. A service fails either because it does not comply with the functional specification, or because this specification did not adequately describe the system function. A service failure is a transition from correct service to incorrect service, i.e., to not implementing the system function. The period of delivery of incorrect service is a **service outage**. The transition from incorrect service to correct service is a **service restoration**. The deviation from correct service may assume different forms that are called **service failure modes** and are ranked according to failure severities.
Failure can arise from anywhere.
The customer does not care.
Failure Viewpoints

- Failure domain
- Detectability of failures
- Consistency of failures
- Consequences of failure
Failure Domain

- Content failure:
  - Supplied service has incorrect information

- Timing failure:
  - Timing or duration of information incorrect
  - Halt failure:
    - System halts and remains in fixed state

- Erratic failure:
  - Service delivery is erratic
Detectability And Consistency

- **Detectability:**
  - Signaled, unsignaled, false alarm
    - False positives, false negatives

- **Consistency:**
  - Consistent—incorrect service seen identically by all users
  - Inconsistent—incorrect service seen differently by different users, *Byzantine* failures
Consequences Of Failure

- Different failures, different consequences

- Examples (go with intuitive notion for now):
  - Availability: Long vs. short outage
  - Safety: Human life endangered vs. equip
  - Confidentiality: Type of information disclosed

- Range of severity of failure:
  - Minor failure
  - Catastrophic failure
Attributes Of Dependability

- Reliability
- Availability
- Safety
- Confidentiality
- Integrity
- Maintainability

Requirements: These are properties that might be required of a given system

Customers must identify the dependability requirements of their system and developers must design so as to achieve them.
Attributes Of Dependability

- Note this very carefully…
- Fault tolerance is *not* a system requirement or a system property
- Fault tolerance is one of the *mechanisms* that can be used to provide dependability
- This is *important*
Reliability

\[ R(t) = \text{Probability that the system will operate correctly in a specified operating environment up until time } t \]

- System’s ability to provide continuous service
- Note that \( t \) is important
- If a system only needs to operate for ten hours at a time, then that is the reliability target
Failure Per Demand

- Sometimes, the notion of time, $t$, is not what we need
- Some systems are only called to act on demand:
  - E.g., Protection systems
- For such systems, what we need is:

  \[ \text{Probability of failure per demand} \]
Availability

\[ A(t) = \text{Probability that the system will be operational at time } t \]

- Literally, readiness for service
- Admits the possibility of brief outages
- Fundamentally different concept
Reliability vs. Availability

- They are not the same.....
- Example:
  A system that fails, on average, once per hour but which restarts automatically in ten milliseconds would not be considered very reliable but is highly available
  Availability = 0.9999972
# Nines of Availability

<table>
<thead>
<tr>
<th># 9’s</th>
<th>%</th>
<th>Downtime / year</th>
<th>Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>99%</td>
<td>~5000 minutes</td>
<td>General web site</td>
</tr>
<tr>
<td>3</td>
<td>99.9%</td>
<td>~500 minutes</td>
<td>Amazon.com</td>
</tr>
<tr>
<td>4</td>
<td>99.99%</td>
<td>~50 minutes</td>
<td>Enterprise server</td>
</tr>
<tr>
<td>5</td>
<td>99.999%</td>
<td>~5 minutes</td>
<td>Telephone System</td>
</tr>
<tr>
<td>6</td>
<td>99.9999%</td>
<td>~30 seconds</td>
<td>Phone switches</td>
</tr>
</tbody>
</table>

Caveats: How measured? What does it mean to be operational?
Design Tradeoffs

Availability = \frac{MTTF}{MTTF + MTTR}

- How to make availability approach 100%?
  - MTTF \rightarrow \infty \text{ (high reliability)}
  - MTTR \rightarrow 0 \text{ (fast recovery)}

- Need to define maximum repair time
- Need to define how Availability is measured
Subtleties

- Which has higher availability?
  - Two 4.5 hour outage / year
  - 1 minute outage / day

- For an Internet-base company such as EBay or Amazon, which would be more desirable? Why?

- For an autonomous rover?

- Need to specify details of acceptable outages
Specifying Availability

- Minimum probability of readiness for service
- Period over which probability measured
- Fixed or moving window
- Maximum acceptable outage
  - Time to repair
- Minimum sustained operating time
Safety

Absence of catastrophic consequences on the users or the environment

- Are commercial aircraft “safe”?  
- They crash occasionally  
- How many crashes are too many?  
- Are cars “safe”? They crash quite a lot

40K deaths/yr; 800/week = 2 fully loaded Boeing 747/week
Risk

- *Risk* is defined to be expected loss per unit time:

\[
\text{Risk} = \sum \text{pr(accident}_i\text{)} \times \text{cost(accident}_i\text{)}
\]

- *Safety is defined to be an acceptable level of risk:*

\[
\text{Risk} < \text{pre-defined}_\delta \text{elta}
\]
Reliability vs. Availability vs. Safety

- They are not the same.....

- Example:
  A system that is turned off
  is not very reliable,
  is not very available,
  but is probably very safe

- In practice, safety often involves specific intervention

- Never believe an airline that says: “Your safety is our biggest concern.”
Confidentiality

Absence of unauthorized disclosure of information

- Microsoft source code vs. Linux source code
- Web browsing
- Operating systems security model
  - Files, memory
- Medical records
- Credit card transaction records
- School grades
Integrity

Absence of improper system state alterations

- Operating systems:
  - Files, memory, network packets
- Linux kernel backdoor attempt
- Database records
- Your bank account
- File transfer
- Did I really get the right version of software XYZ?
- …
Security

Security is a combination of attributes:
- Integrity
- Confidentiality
- Availability

Under different circumstances, these attributes are more or less important:
- Denial of service is an availability issue
- Exposure of information is a confidentiality issue
Maintainability

Ability to undergo repairs and modifications
General Dependability Requirements

- Telecommunications:
  - Availability, maintainability

- Transportation:
  - Reliability, availability, safety

- Weapons:
  - Safety

- Nuclear systems:
  - Safety
Pacemaker Dependability

Dual-Chamber Pacemaker

Pacing now extended to left ventricle
General Characteristics

- Eight-bit processors, moving to 32-bit
- Software:
  - Approximately 30K lines, mostly “C”
  - Vastly more software in external programmer
- Patient data storage example:
  - 200 samples/sec, 3 channels, 15 minutes
- Long battery life necessary—device “sleeps” between heart beats
Pacemaker Inputs & Outputs

- Three leads—one from each of three chambers (L, R ventricles, R atrium)
- Each lead can sense and pace
- Two shocking electrodes in right ventricle
- Case to lead impedance for breathing rate and lung volume
- Accelermeter to measure activity
- Body temperature
Real-time Control

Timer Interrupt → Heart Beat

Wake up

Check Expected Event

Event As Expected?

Yes → Set Timer

No → Generate Signal
Multi-Chamber Functionality

- **Pacing:**
  - Always allow natural pace if possible
  - Atrium-ventricle sequencing to allow for fill
  - Atrium natural pulse, pace ventricle
  - Biventricular pacing for congestive heart failure

- **Defibrillation:**
  - Ventricular fibrillation relatively easy to detect
  - Distinguish benign, moderate, serious events
  - Atrial fibrillation being treated also
Dependability of Pacemaker

- Is reliability the goal?
- Typical battery life is five years
- Does failure matter?
  - Device is “off” between heartbeats
  - 70 Hz restart rate is acceptable
  - So $t = 100$ milliseconds is OK
- But persistent storage is needed
Dependability of Pacemaker

- Is availability the goal?
- How about an availability of 0.99999?
- This corresponds to an average of five minutes per year of downtime
- Death would result if this occurred all at once
Dependability of Pacemaker

- Is safety the goal?
- It’s safe when it’s off—or is it?
- Leaving the system off might result in death very quickly.....

- Safety:
  - Under/over pacing
  - Unnecessary/missing defibrillation
Attributes Of Dependability

- Reliability
- Availability
- Safety
- Confidentiality
- Integrity
- Maintainability

Requirements: These are properties that might be required of a given system.

Customers must identify the dependability requirements of their system and developers must design so as to achieve them.
How Much Is Enough?

- Dilemma—could we achieve higher dependability:
  
  *If more resources were expended on developing a given system then the rate of failure might be reduced. That being the case, should the developers expend all possible resources on developing the system?*

- For example, formal verification:
  
  - Known to detect defects
  - Considered difficult and expensive to apply
  - Should its use be required of all systems with large consequences of failure?
ALARP

- As Low As Reasonably Practicable

- Three degrees of risk:
  - Tolerable (broadly acceptable)
  - Unacceptable—has to be reduced
  - ALARP region

Carrot Diagram